Faculty of Mathematics, Natural Sciences and Computer Science

Institute of Computer Science

COMPUTER SCIENCE REPORTS

Report 01/13

April 2013

OVA-BASED MULTI-CLASS CLASSIFICATION FOR DATA STREAM ANOMALY DETECTION

TINO NOACK
INGO SCHMITT
SASCHA SARETZ

Computer Science Reports
Brandenburg University of Technology Cottbus
ISSN: 1437-7969

Send requests to: BTU Cottbus
Institut für Informatik
Postfach 10 13 44
D-03013 Cottbus

Tino Noack, Ingo Schmitt, Sascha Saretz
noacktin@tu-cottbus.de, schmitt@tu-cottbus.de, saretsas@tu-cottbus.de,
http://dbis.informatik.tu-cottbus.de

OVA-based Multi-Class Classification for Data Stream Anomaly Detection

Computer Science Reports
01/13
April 2013

Brandenburg University of Technology Cottbus

Faculty of Mathematics, Natural Sciences and Computer Science

Institute of Computer Science

Computer Science Reports
Brandenburg University of Technology Cottbus
Institute of Computer Science

Head of Institute:
Prof. Dr. Ingo Schmitt schmitt@tu-cottbus.de
BTU Cottbus
Institut für Informatik
Postfach 10 13 44
D-03013 Cottbus

Research Groups:	Headed by:
Computer Engineering	Prof. Dr. H. Th. Vierhaus
Computer Network and Communication Systems	Prof. Dr. H. König
Data Structures and Software Dependability	Prof. Dr. M. Heiner
Database and Information Systems	Prof. Dr. I. Schmitt
Programming Languages and Compiler Construction	Prof. Dr. P. Hofstedt
Software and Systems Engineering	Prof. Dr. C. Lewerentz
Theoretical Computer Science	Prof. Dr. K. Meer
Graphics Systems	Prof. Dr. D. Cunningham
Systems	Prof. Dr. R. Kraemer
Distributed Systems and Operating Systems	Prof. Dr. J. Nolte
Internet-Technology	Prof. Dr. G. Wagner

CR Subject Classification (1998): H, H.2.8

Printing and Binding: BTU Cottbus

ISSN: 1437-7969

OVA-based Multi-Class Classification for Data Stream Anomaly Detection

Tino Noack, Ingo Schmitt, Sascha Saretz

Brandenburg University of Technology
Cottbus, Germany
Institute of Computer Science, Information and Media Technology
{Tino.Noack, Ingo.Schmitt, Sascha.Saretz}@tu-cottbus.de

Abstract. Mobile cyber-physical systems (MCPSs), such as the International Space Station, are equipped with sensors which produce sensor data streams. Continuous changes like wear and tear influence the system states of a MCPS continually during runtime. Hence, monitoring is necessary to provide reliability and to avoid critical damage. Although, the monitoring process is limited by resource restrictions. Therefore, the focal point of the present paper is on time-efficient multi-class data stream anomaly detection. Our contribution is bifid. First, we use a one-versus-all classification model to combine a set of heterogeneous one-class classifiers consecutively. Such a chain of one-class classifiers provides a very flexible structure while the administrative overhead is reasonably low. Second, based on the classifier chain, we introduce classifier pre-selection.

1 Introduction

Mobile cyber-physical systems (MCPSs), such as the International Space Station (ISS), are location independent and embedded into a physical environment. Mechanical influences (e.g. continual friction) as well as external impacts of a harsh and uncertain physical environment (e.g. geophysical effects) can cause wear and tear. Mostly, wear and tear leads to system state changes which can cause sudden changes such as crashes. On that score, monitoring MCPSs is indispensable to ensure reliability and to avoid critical damage. The monitoring process, which takes place on a MCPS, is often limited by resource restrictions (e.g. processing capacity, memory or power consumption). To compensate resource restrictions an external wireless network often connects a MCPS with external information systems. External information systems are usually stationary parts of a MCPS.

MCPSs are equipped with sensors which produce sensor data streams [1, 12]. These data streams have to be processed in an appropriate manner for monitoring MCPSs. Further information about real-time monitoring are provided by [33]. Data stream processing [7, 28], which involves knowledge discovery from data streams (KDDS) [11] as well as data stream mining [4], has received much attention in recent years. Substantial components of data stream processing are data stream clustering, data stream classification and data stream anomaly detection. The time-efficiency is very important for data stream processing. However, there

exists only a little work considering time-efficient data stream anomaly detection at the moment. Applying anomaly detection techniques in a data stream context is necessary to monitor MCPSs; and simultaneously, to identify a large number of system states during runtime. For that reason, the focal point of the present paper is on time-efficient multi-class data stream anomaly detection.

1.1 Basic Notation

As depicted in Figure 1, a multi-class anomaly detection problem is based on a set of classes (bounded regions) $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$. These classes represent expert knowledge about the system states of a MCPS. A vector space S [2] and the cluster assumption [25] are the basic foundations. A multi-class anomaly detection problem corresponds to the anomaly class $\Omega^{\mathrm{C}} = S \setminus \bigcup_i \omega_i$. The anomaly class represents the unawareness about the system states. A data stream is considered as a continuous and almost infinite stream of unlabelled instances [1]. A vector space S is spanned by a set of n mutually independent (orthogonal) attributes $A_1, \ldots, A_n$. Attribute values are functions over time T, i.e. values of A_i with $i = \{1 \,..\, n\}$ are values of $a_i : T \to \mathbb{R}$. We denote the time as an index. Hence, an unlabelled instance with index t is represented as $\boldsymbol{s}_t = (a_{t,1}\ a_{t,2}\ ...\ a_{t,n})' \wedge \boldsymbol{s}_t \in S$.

1.2 Problem Description

The shape of the present classes can diverge widely. Hence, an approximation of the present multi-class anomaly detection problem is required. We suggest to use a distinct and optimized classifier for each class. Therefore, a multi-class anomaly detection problem can be comprehended as a set of binary decisions. Each binary decision corresponds to a *dichotomous classifier*. A dichotomous classifier provides two mutually exclusive decisions. It returns *true* if an unlabelled instance was accepted and *false* if an unlabelled instance was not accepted. At most one of a set of dichotomous classifiers yields true. This implies a distinct classification result. A set of dichotomous classifiers entails two advantages in contrast to homogeneous multi-class classification. On one hand, such a set of classifiers usually provides a higher classification accuracy. On the other hand, a multi-class anomaly detection problem falls apart into a set of simple binary classification problems.

The set of dichotomous classifiers have to be combined in a sufficient manner. A combination is required to solve a multi-class anomaly detection problem under timing-constrains. We suggest a consecutive application of these dichotomous classifiers. The resulting classifier chain provides a very flexible structure while the administrative overhead is reasonably low. Moreover, we suggest to use an *one-versus-all* (OVA) [13, 14, 31] multi-class classifier model. Each classifier of an OVA-based multi-class classification model is trained between the target class and the other classes including the anomaly class. Therefore, we suggest the application of a set of one-class classifiers [30]. Versatile performance studies of one-class classifiers are contributed by [5, 15]. As stated in [30], each one-class

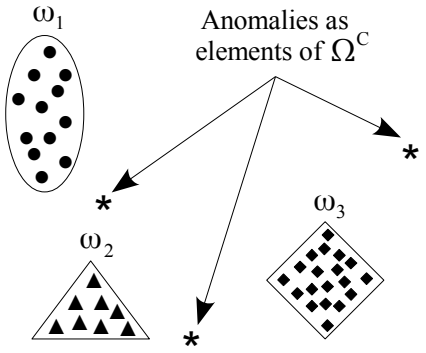

Fig. 1. Example of a multi-class anomaly detection problem (based on [6])

classifier algorithm comprises two distinct elements. First, a distance or a resemblance of an instance to a class. Second, a threshold θ on this distance or resemblance. This threshold is used to define a one-class classifier as a dichotomous classifier. Further on, a chain of disjointed one-class classifiers provides a distinct classification result for a multi-class anomaly detection problem.

To the best of our knowledge no substantial research has been conducted to study OVA-based multi-class classification models for data stream anomaly detection. Due to the consecutive application of one-class classifiers, an OVA-based multi-class classification model becomes to a linear or sequential search problem. Further information about sequential searching are provided by [16]. The main disadvantages of such a chain of one-class classifiers are time-inefficiencies. However, the reduction of processing time can also help to reduce power consumption. Therefore, our contribution is a minimization of the average processing time. A one-class classifier entails a probability of occurrence and a processing time. The absolute processing time depends on a specific target machine. First, we minimize the expected value over the probability of occurrence in conjunction with the processing time of the component one-class classifiers. This minimization is intended to decreases the processing time of the classifier chain for the majority of unlabelled instances. Second, we minimize the processing time in the worst case scenario when an unlabelled instance has to be processed by all of the component classifiers. Therefore, we suggest classifier pre-selection. We approximate each one-class classifier by means of a minimal bounding hypersphere. Afterwards, the distances of an unlabelled instance to all of the hypersphere centres can be calculated. On that score, it is possible to preselect a reduced set of candidate classifiers. Moreover, we use the ISS Columbus Air Loop [19, 20] in a real world case study to evaluate and assess the aforementioned suggestions.

The remaining sections of the paper are structured as follows. We discuss related work in Section 2 and Section 3 introduces the ISS Columbus Air Loop. Moreover, we describe the minimization of the average processing time of a

classifier chain in Section 4 and classifier pre-selection is delineated in Section 5. We explain the experimental setup and perform the real world case study in Section 6. Finally, we discuss the results and conclude our work in Section 7.

2 Related Work

Commonly, classification-based anomaly detection encompasses three phases: *training*, *assessing* and *classifying*. Training is used to learn a classifier model from a set of labelled training data $\mathcal{X}^{tr} \subset S$. Assessing is used to evaluate and refine the learned model by training instances (e.g. cross validation [10]). Classifying is used to classify unlabelled instances by the learned model. Classification-based anomaly detection is grouped into two categories: *multi-class* and *one-class* classification techniques. As stated in [6], multi-class classification-based anomaly detection works mostly in a supervised manner. It presumes that the training data $\mathcal{X}^{tr}$ involve multiple classes which include the anomaly class. One-class classification-based anomaly detection works in a semi-supervised manner. It presumes that the training data involve only one class label [6, 30].

Normally, monitoring MCPSs requires the interaction with human experts during the training and assessing phases. For example, the ISS Columbus Failure Management System [19, 20] dispatches the occurring data streams almost completely (down-sampled) to an external information system (the ground station) to provide historical data. This contrasts the widely adopted assumption that data streams cannot be stored almost completely. Indeed, such an almost complete storage is very expensive. But among other reasons, human experts are responsible for related and consequent decisions. Therefore, historical data is a prerequisite for long-term failure analysis and adequate anomaly detection.

Considering the existence of an external information system, we distinguish between *online* and *offline* training methods for data stream anomaly detection. Online training methods neglect the existence of external information systems and they assume that a data stream cannot be stored completely. Furthermore, they provide the ability for training the model; and simultaneously, for classifying unlabelled instances in real-time or near real-time. For this purpose, online training methods are applied by means of one-pass algorithms while only a small window-based set of training instances is available. In the context of online data stream anomaly detection, user interaction and fast algorithms are mutually exclusive. Therefore, the assessment of the resulting model is often neglected by online training methods. One reason is the absence of a reasonable number of training instances due to windowing. Hence, the accuracy is proved insufficiently. Besides, the verification of classifying results by human experts is very difficult. Some online training methods require an input of labelled training instances during runtime and at certain time intervals for model training or retraining. The OcVFDT [18, p. 80] approach, for example, prerequisites 20% of labelled training data during runtime. Such an online training method entails three more drawbacks. First, the provision of labelled training data during runtime cannot be always guaranteed and it contradicts the functioning of most of the real world

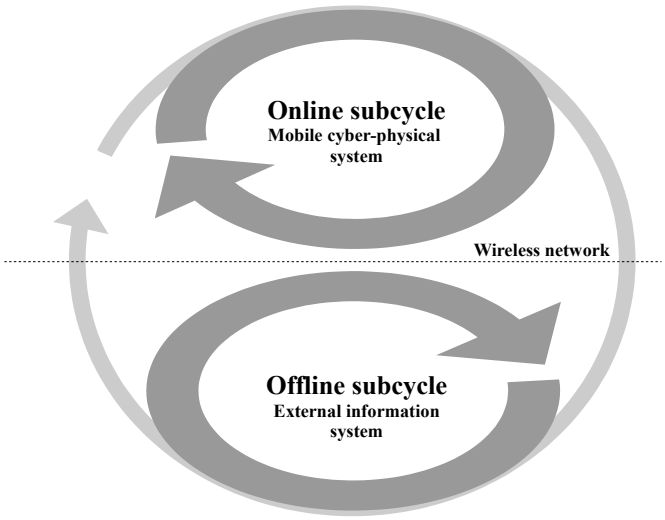

Fig. 2. KDC - A cyclic monitoring process (based on [21, 22])

applications. Second, model training or retraining during runtime expends computational resources which are actually envisaged for system monitoring. Third, training such an unassessed model during runtime can cause unforeseeable and critical side effects for the entire monitoring process.

Due to the aforementioned drawbacks, our work focuses on offline training methods. As depicted in Figure 2, offline training methods require an *offline subcycle* and an *online subcycle* of a Knowledge Discovery Cycle (KDC) [21, 22]. The offline subcycle is used for long-term analysis, for model training and model assessing. Model training and model assessing are performed on an external information system which provides much system resources. Thus, heavyweight algorithms can be applied while historical training instances are available. Therefore, training the model can be very time consuming while the online resources are not adversely affected and the resulting model can be easily assessed by human experts. The trained model needs to be transferred from the offline subcycle to the online subcycle from time to time. The online subcycle is used to apply the trained model for data stream anomaly detection. Computational resources of the online subcycle are exclusively used for classifying unlabelled instances by means of the previously created model. Therefore, classifying can be very fast. Following this, we discuss present algorithms for data stream anomaly detection.

OLINDDA [27] implements a cluster-based approach for novelty and concept drift detection. By default, OLINDDA works in an unsupervised manner and it uses standard clustering algorithms to identify unknown clusters. Afterwards, the similarity to known clusters is assessed. As stated in [29, p. 1512], this method detects new emerging concepts rather than anomalies. Thus, OLINDDA

is not generally used for the purpose of data stream anomaly detection which constitutes the main disadvantage of OLINDDA.

FRAHST [24] is a rank-adaptive algorithm for fast principal subspace tracking. It works in an unsupervised manner and it is used to identify anomalies in streaming data of low dimensions. Therefore, the subspaces are built using dimensionality reduction. Observed data that cannot be sufficiently explained by the current model is considered as anomalous. One obvious disadvantage of this method is the loss of information during dimensionality reduction. Moreover, it does not provide the ability to distinguish between heterogeneous classes.

A method which uses ensembles of streaming half-space trees (HS-Trees) is described in [29]. A HS-Tree is a binary tree while each node is used to capture a number of data elements within a particular subspace of the data stream. The HS-Trees method is a fast one-class anomaly detector for evolving data streams. As stated in [29, p. 1511], the HS-Trees approach requires only 'normal' data, which excludes the anomaly class, to retrain the model. The model is retrained continuously at the end of each window. The main disadvantage of the proposed method is the paraxial separation of the subspaces. Hence, the anomaly detection task becomes ambiguous when the subspaces are not paraxially separable.

A very fast decision tree for one-class classification of data streams (OcVFDT) is described in [18]. This OcVFDT approach is an extension of the very fast decision tree (VFDT) [9]. During the training phase, the OcVFDT algorithm constructs a tree forest and then the best tree is chosen as final output classifier. As stated in [18, p. 80], OcVFDT presupposes approximately 20% of labelled training instances during runtime to retrain the classification model. The proposed method contains two disadvantages. First, it uses discrete attribute values. Hence, it is not widely applicable. Second, the computational effort of this method depends on the selected one-class classification algorithm. Thus, training a tree forest as well as the classification of unlabelled instances can be very time-consuming when a complex one-class classifier has been chosen.

We analysed four data stream anomaly detection methods. The assessment and the evaluation of the resulting model by human experts is neglected by all of the analysed methods. Therefore, training a solid model, which is based on long-term historical data, is not sufficiently considered by the analysed methods. Considering these disadvantages, our contribution is the adjustment of an offline training method, in particular OVA-based multi-class classification, for anomaly detection to the demands of data streams under timing constrains.

3 Example - ISS Columbus Air Loop

The ISS Columbus Module [19, 20] is a MCPS and it comprises an air loop. The ISS Columbus Air Loop is part of the life support system. The ISS Columbus Failure Management System is responsible for crew health and for detecting time critical failures. The ISS Columbus Air Loop is monitored by the failure management system. The air loop consists of fan assemblies which provide air circulation in the ISS Columbus Module. Moreover, the fan assemblies are responsible for

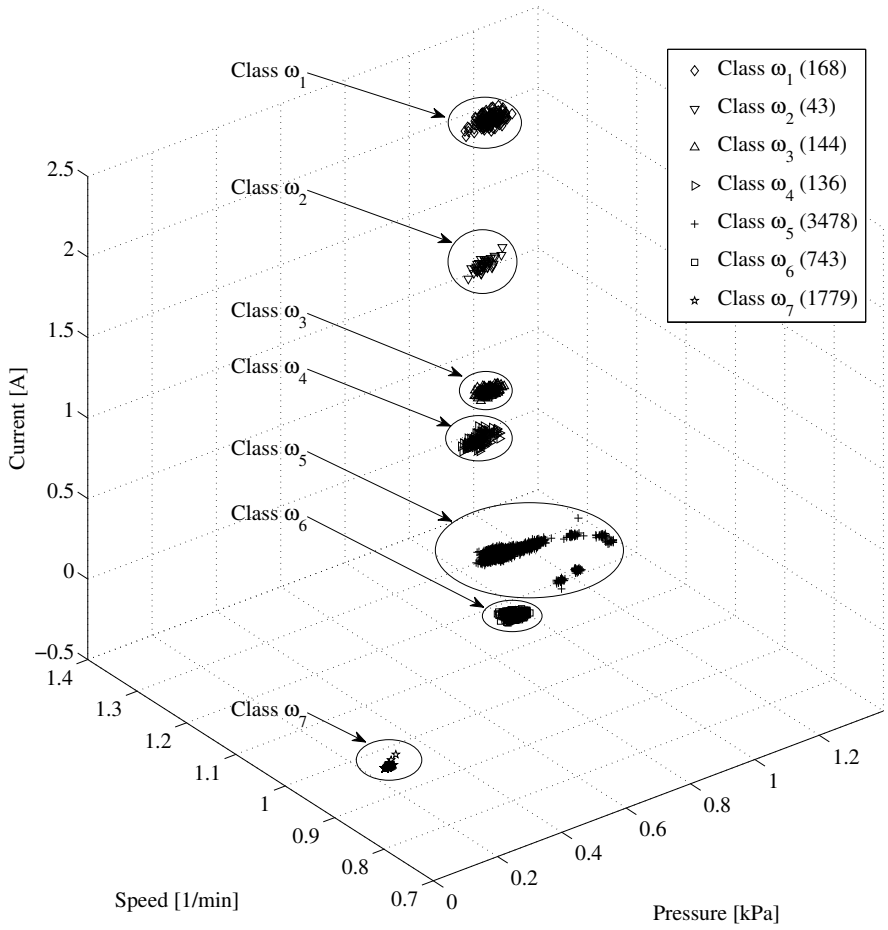

Fig. 3. IRFA training data and seven classes

air exchange between the ISS Columbus Module and the ISS. Amongst others, air circulation is required for air revitalization, for air conditioning, for smoke or fire detection and for avoiding dead air pockets. The ISS Columbus air loop contains an **I**nter Module Ventilation **R**eturn **F**an **A**ssembly (IRFA). The IRFA is monitored by different sensors. For the sake of simplicity, we focus on three sensor attributes. This includes *speed*, *current* and *pressure*. The speed relates to the rotating speed of the fan assembly and the unit of measurement is $1/min$. The current relates to the electrical input current of the IRFA and the unit of measurement is A. The pressure relates to the pressure head that is generated by the IRFA and the unit of measurement is kPa. Figure 3 depicts the training data which was previously clustered by human experts. The training data comprise seven classes $\Omega = \{\omega_1, \omega_2, ..., \omega_7\}$. Each class refers to a specific system

state of the IRFA. The class ω_1, for example, relates to a faulty system state while the speed of the IRFA is unusually increased. The class ω_5, for example, refers to a default system state while the IRFA works as expected. The class ω_7, for example, describes a system state while the IRFA has been switched off. The values in the parentheses denote the number of instances.

4 Minimizing the Time Consumption in Average

Figure 4 depicts a consecutive chain of a set of one-class classifiers $occ_{(i)}$. Each one-class classifier is a dichotomous classifier. Moreover, each one-class classifier provides a binary decision (true or false) due to the provided threshold θ. True implies that an unlabelled instance was accepted by a classifier and false implies the opposite. We define the applied one-class classifiers as disjoint to avoid ambiguous classification results. Therefore, the combination of these mutually exclusive classifiers leads to a distinct classification result. The consecutive chain of one-class classifiers refers to OVA-based multi-class classification while the classifiers are combined by means of binary decisions. The applied one-class classifiers can be rearranged into $k!$ many different permutations π.

The *termination condition* of such an OVA-based multi-class classifier model entails two possibilities. First, the algorithm terminates when an unlabelled instance was accepted by a classifier. Second, the algorithm terminates when all of the component one-class classifiers fail. The worst case scenario relates to a case when an unlabelled instance has to be processed by all of the component one-class classifiers. Each one-class classifier entails a processing time $t_{(i)}$. The bracketed indexes refer to a permutation. Amongst others, Bifet et al. established a requirement for data stream environments which states: "Process an example [an unlabelled instance] at a time, and inspect it only once (at most)" [3, p. 1601]. In conformity with this statement, the processing time of a one-class classifier refers to the classification of one unlabelled instance at a time. The main purpose is the minimization of the overall processing time which is consumed by an OVA-based multi-class classifier model to solve a multi-class anomaly detection problem. Hence, it is intended to reach the termination condition as fast as possible for the majority of the unlabelled instances in average.

Under consideration of a set of training data $\mathcal{X}^{tr}$ it is possible to estimate the probabilities of occurrence for each class p_i, where $p_1 + p_2 + \cdots + p_k = 1$. These probabilities can be assigned to the corresponding classifiers.

$$p_i = \frac{|\{s_t \in \mathcal{X}^{tr} \wedge s_t \in \omega_i\}|}{|\mathcal{X}^{tr}|} \tag{1}$$

Based on these estimates it is possible to select a permutation from the set of all permutations such that the termination condition is reached as early as possible for the majority of the unlabelled instances. According to [16, p. 399], this can be achieved when the one-class classifiers are sorted descending by the estimated probabilities.

$$p_1 \geq p_2 \geq \cdots \geq p_k \tag{2}$$

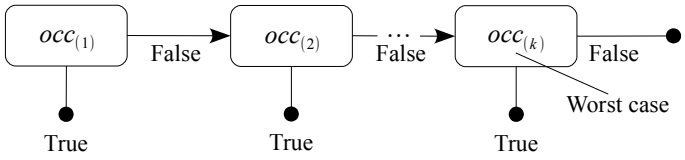

Fig. 4. One-versus-all multi-class classifier chain

The main disadvantage is that the processing time was not taken into account. Therefore, we suggest to consider the probability of occurrence as product with the processing time. Because of the consecutive chain of the one-class classifiers, the processing time $\tilde{t}_{(i)}$ of a classifier is the sum of the previously executed classifiers. This refers to the expected value μ_π of a permutation π.

$$\tilde{t}_{(i)} = \sum_{m=1}^{i} t_{(m)},\ \mu_{\pi_l} = \sum_{m=1}^{k} p_{(m)} \cdot \tilde{t}_{(m)} \tag{3}$$

The permutation with the minimal expected value provides the minimal average processing time. The empirical calculation of the minimal expected value represents a feasible solution. However, this solution is very expensive due to the calculation of $k!$ many permutations. Therefore, we introduce the following theorem which is based on [26] and [16, p. 404].

Theorem 1. *Let $p_{(i)}$, $t_{(i)}$ and $\tilde{t}_{(i)}$ be as defined above. The arrangement of the one-class classifiers in an OVA-based multi-class classification model is optimal if and only if*

$$\frac{p_{(1)}}{t_{(1)}} \geq \frac{p_{(2)}}{t_{(2)}} \geq \cdots \geq \frac{p_{(k)}}{t_{(k)}}. \tag{4}$$

In other words, the minimal expected value over all permutations π provides the minimal average processing time if and only if (4) holds.

Proof. Suppose that $p_{(i)} \cdot \tilde{t}_{(i)}$ and $p_{(i+1)} \cdot \tilde{t}_{(i+1)}$ are interchanged. On that score, a permutation changes from

$$\cdots + p_{(i)} \cdot \tilde{t}_{(i)} + p_{(i+1)} \cdot \tilde{t}_{(i+1)} + \cdots \tag{5}$$

to

$$\cdots + p_{(i+1)} \cdot (\tilde{t}_{(i+1)} - t_{(i)}) + p_{(i)} \cdot \tilde{t}_{(i+1)} + \cdots . \tag{6}$$

This results in a change of the expected processing time by $p_{(i+1)} \cdot t_{(i)} - p_{(i)} \cdot t_{(i+1)}$. Therefore under the given assumptions, it follows that the change from (5) to (6) increases the processing time. Consequently, the permutation (6) is not optimal and (4) holds for any optimal permutation.

According to [16, p. 404], we showed that the permutation which provides the minimal expected value is locally optimal and that adjacent interchanges lead to

no further improvements. Moreover, it is necessary to show that the permutation is globally optimal. As stated in [16, p. 404], we consider two proofs. The first proof uses computer science and the second proof uses a mathematical trick. Finally, we consider three special cases.

First proof. Assume that (4) holds and consider that the one-class classifiers are sorted as follows $occ_{(1)}, occ_{(2)}, ..., occ_{(k)}$. Such an arrangement can be achieved by using a sequence of interchanges such that each interchange replaces $..., occ_{(j)}, occ_{(i)}, ...$ by $..., occ_{(i)}, occ_{(j)}, ...$ for some $i < j$. This decreases the overall processing time in average by the non-negative amount $p_{(i)} \cdot t_{(j)} - p_{(j)} \cdot t_{(i)}$. Thus, the permutation which provides the minimal expected value also provides the minimal average processing time.

Second proof. In accordance with [16, p. 404], replace each probability $p_{(i)}$ by

$$p_{(i)}(\epsilon) = p_{(i)} + \epsilon^i - (\epsilon^1 + \epsilon^2 + \cdots + \epsilon^k)/k, \tag{7}$$

where ϵ is a really small positive number. In the case that ϵ is sufficiently small we can exclude $x_1 p_{x1}(\epsilon) + \cdots + x_k p_{xk}(\epsilon) = y_1 p_{y1}(\epsilon) + \cdots + y_k p_{yk}(\epsilon)$ unless $x_1 = y_1, ..., x_k = y_k \wedge p_{x1}(\epsilon) = p_{y1}(\epsilon), ..., p_{xk}(\epsilon) = p_{yk}(\epsilon)$. Therefore, (4) will not hold if the processing times of all one-class classifiers are equal and the probabilities of occurrence of all one-class classifiers are equally distributed as well. This contrasts the proof of [16, p. 404]. The proof of [16, p. 404] demands the inequality of only one parameter $x_1 \neq y_1, ..., x_k \neq y_k$. On that score, the proof of [16, p. 404] neglects the interchanging of the probabilities of each permutation. This contradicts the fundamental assumption [16, p. 399] while (2) holds.

Under consideration of all $k!$ permutations of the component one-class classifiers, we know that there exist at least one permutation which satisfies (4) due to the exclusion of equality of both parameters (processing time and probability of occurrence). Hence, (4) uniquely determines the permutation with the minimal expected value for the probabilities $p_i(\epsilon)$ if ϵ is sufficiently small. By continuity, this also holds if ϵ is set equal to zero. Following, we consider three special cases.

(a.) Special case one. The first special case takes into account that the processing times of all component one-class classifiers are equal $t_{(1)} = t_{(2)} = \cdots = t_{(k)}$. Therefore, the processing time can be reduced while (2) and (4) hold. Hence, the one-class classifiers are arranged in descending order by the probabilities. This refers to the basic assumption of [16, p. 399] while (2) holds.

(b.) Special case two. The second special case takes into account that the probabilities of occurrence are equally distributed $p_{(1)} = p_{(2)} = \cdots = p_{(k)}$. Therefore, the probabilities can be reduced and (4) holds. Thus, the one-class classifiers are arranged in ascending order by the processing times.

(b.) Special case three. The third special case refers to both previous special cases while the processing times are equal and the probabilities of occurrence are equally distributed as well. Hence, the expected values of all permutations are equal and it exists no permutation with minimal expected value as claimed by the theorem. Hence, the theorem does not hold for equality of both parameters.

□

5 Classifier Pre-Selection

The application of an OVA-based multi-class classification model for data stream anomaly detection leads to a chain of one-class classifiers. The previous section discusses the minimization of the processing time for the majority of unlabelled instances. Although the worst case scenario, when all component one-class classifiers have to be processed, is still remaining. The chain of one-class classifiers provides a very flexible structure and it can be used to take further advantages of it. The main idea of the suggested classifier pre-selection is the extraction of a set of candidate classifiers such that the processing time can be further decreased. Therefore, we suggest to approximate each one-class classifier with an additional classifier. This leads to an ensemble of two classifiers for each class. Ensembles of classifiers are aimed to create more accurate classification decisions by combining classifiers for a given classification problem at the expense of increased complexity [8, 11, 17]. However, we use an ensemble of two classifiers to reduce the processing time by classifier pre-selection.

As depicted in Figure 5, we suggest to use a minimal bounding hypersphere h_i to represent this additional classifier. Each hypersphere bounds the corresponding one-class classifier minimally. Moreover, each hypersphere provides a centre c_i and a radius r_i. In contrast to all other geometrical descriptions, a hypersphere provides a material advantage. The distance of an unlabelled instance to the centre of a hypersphere is independent from the position of the unlabelled instance to the hypersphere. Contrary to the one-class classifiers, the hyperspheres must not be disjoint. If an unlabelled instance $\boldsymbol{s}_t$ needs to be classified, the distances to all of the centres of the hyperspheres $d_i(\boldsymbol{s}_t, c_i)$ have to be calculated. A one-class classifier is a candidate if the distance of an unlabelled instance to the centre is below the radius of the bounding hypersphere $d_i(s_t, c_i) \leq r_i$. The resulting set of candidate classifiers is still in an optimal arrangement while the remaining classifiers can be excepted.

We assume that the calculation of the distances is less time consuming than the calculation of all component one-class classifiers. This assumption holds for use cases when the number of classes is relatively low and the processing time of the one-class classifiers is relatively high. However, when the number of one-class classifiers increases, the calculation of the distances increases as well. On that score, we suggest to use the triangle inequality to approximate the distance of an unlabelled instance to a class. This decreases the processing time additionally.

6 Experiments

At the beginning, this section describes the experimental setup. Further on, we present the minimization of the time consumption in average and classifier pre-selection. Finally, we compare our suggestions with the HS-Tree approach [29].

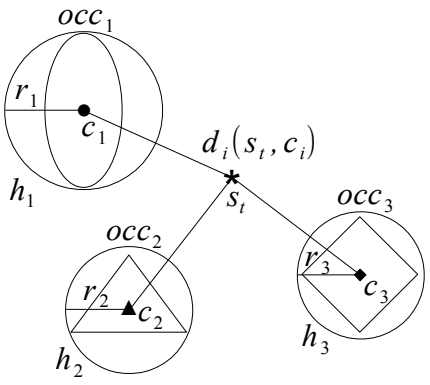

Fig. 5. Classifier pre-election - Ensemble of two classifiers for each class

6.1 Experimental Setup

As depicted in Figure 6, our implementation provides an offline subcycle and an online subcycle which are based on the KDC [21, 22]. The offline subcycle is represented by a client and the online subcycle is represented by a server. The client and the server are weakly coupled whereas the communication takes place by means of an external network. The offline subcycle is used to train the classification model. The training instances are stored in a database (PostgreSQL). Preprocessing, clustering and model training can be realized by means of data mining tools such as Matlab or specific implementations (e.g. Java-based implementations). Thereafter the trained models are stored into a database as well. The communication between the client and the server is implemented my means of a protocol. The client provides a Java-based implementation to retrieve the trained models from the database and to register these models onto the server by means of the protocol. Moreover, the client provides the functionality to generate a data stream. Therefore, unlabelled instances are retrieved from the database and sent to the server using the implemented protocol as well. We used an off-the-shelf computer system (Intel Core i3 with 2.26 GHz and 4 GB memory) for the client.

The server provides the counterpart of the protocol implementation. When the server was started it awaits the handshake with a client. If the communication has been established the server waits for the registration of a data stream, scheduler and anomaly detection algorithms. The scheduler is used to manage the registered anomaly detection algorithms. Finally, when all necessary parts have been registered, the server awaits the initialization of the data stream. The anomaly detection algorithms produce a result for each incoming unlabelled instance. These results are stored into result sets. These result sets are sent to the client by means of the protocol. The retrieved result sets are stored into the data base as well. The server is implemented by means of Java and we used the

Raspberry Pi [23] as target machine which provides a low budget ARM processor (700 MHz with 512 MB memory).

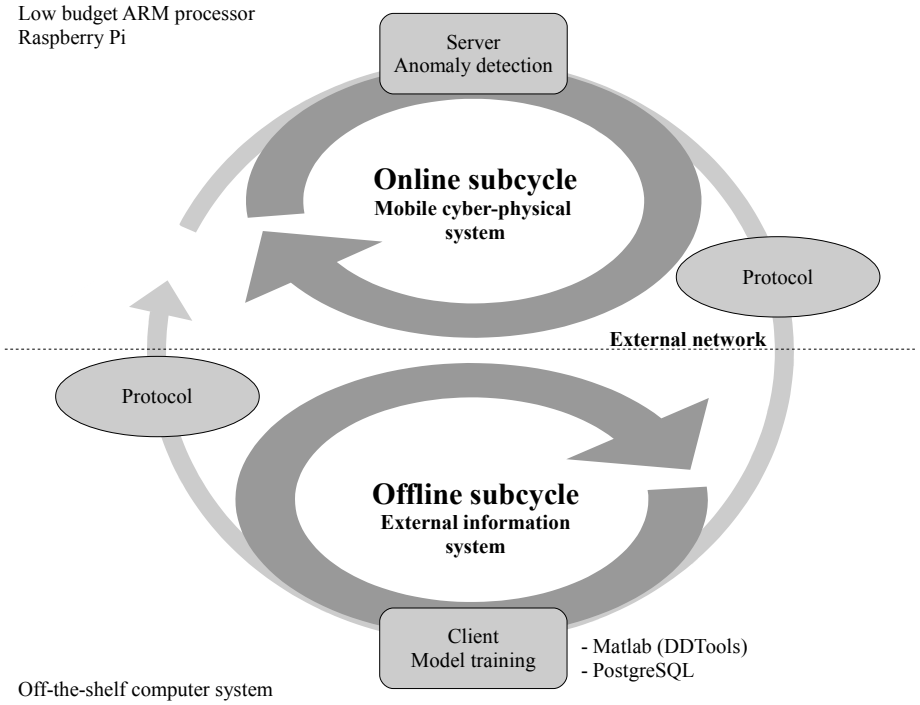

Fig. 6. Prototypical implementation

6.2 Average Time Consumption and Classifier Pre-Selection

According to the aforementioned example, we trained a one-class classifier per class. This includes two Gaussian one-class classifiers, two nearest neighbour (NN) one-class classifiers, a k-centres one-class classifier and two support vector domain description (SVDD) one-class classifiers. Table 1 summarizes the classifier labels and the processing time of each one-class classifier. The one-class classifiers were trained by means of Matlab and the DDtools toolbox [32]. The resulting one-class classifiers were also evaluated by means of a 10-fold cross validation. We determined the processing times in average of the component one-class classifiers empirically (wall-clock time). As presented in Table 1, the processing times of the Gaussian one-class classifiers are almost the same. The processing times of the NN one-class classifiers or the SVDD one-class classifiers differ significantly. The processing time of a NN one-class classifier increases with

the number of training instances. The processing time of the SVDD one-class classifier increases with the number of selected support vectors.

Table 1. Classifier labels and processing times

Label	occ_1	occ_2	occ_3	occ_4	occ_5	occ_6	occ_7
Classifier	Gaussian	NN	k-centres	NN	SVDD	SVDD	Gaussian
$t_{(i)}$ in ms	0.411	3.773	1.703	10.618	3.198	2.113	0.436

We selected two permutations. Permutation π_1 represents the minimized average processing time while permutation π_2 is randomly selected. Moreover, we used two data sets. The first data set $\mathcal{X}_1^{test}$ represents the aforementioned classes with a percentage distribution of the training instances without anomalies. The second data set $\mathcal{X}_2^{test}$ extends the first data set with anomalies. As presented in Table 2, the empirical values are higher than the calculated values due to the administrative overhead. Furthermore, the values are higher when the second data set was used. The reason for this is that the worst case scenario has occurred more often due to the existence of anomalies. Table 2 shows that the processing time of permutation π_1 is under the processing time of permutation π_2. This underscores the aforementioned theorem (Section 4) empirically.

Table 2. Average time consumption

Permutation	Arrangement	μ_{π_l} in ms	$\mathcal{X}_1^{test}$ in ms	$\mathcal{X}_2^{test}$ in ms
π_1	$\omega_7, \omega_5, \omega_1, \omega_6, \omega_3, \omega_4, \omega_2$	3.58	6.14	7.88
π_2	$\omega_4, \omega_5, \omega_7, \omega_6, \omega_1, \omega_3, \omega_2$	14.39	17.31	18.26

Further on, we used the classifier pre-selection to further decrease the processing time in the worst case scenario. Therefore, we calculated the distances from the unlabelled instances to the centres of the hyperspheres directly. However, the triangle inequality could be used to decrease the processing time additionally. Table 3 summarizes the results under consideration of the aforementioned permutations π_1 and π_2 as well as the data sets $\mathcal{X}_1^{test}$ and $\mathcal{X}_2^{test}$. As can be seen, the classifier pre-selection decreases the processing time in each of the cases while the empirically calculated values are approaching each other. We assumed a major difference between both values of π_1 and π_2 for the second data set due to the minimization of the average processing time. We expected that the classifier pre-selection, as extension of the OVA-based multi-class classification model, decreases the processing time additionally. Although in the given data set, the hyperspheres does not intersect. On that score, the arrangement of the one-class classifiers is not evident. We assume that the optimal arrangement becomes more effective if the hyperspheres would intersect.

Table 3. Classifier pre-selection

Permutation	Arrangement	μ_{π_l} in ms	$\mathcal{X}_1^{test}$ in ms	$\mathcal{X}_2^{test}$ in ms
π_1	$\omega_7, \omega_5, \omega_1, \omega_6, \omega_3, \omega_4, \omega_2$	3.58	6.09	6.02
π_2	$\omega_4, \omega_5, \omega_7, \omega_6, \omega_1, \omega_3, \omega_2$	14.39	6.37	6.22

6.3 Comparison with the HS-Tree Approach

A comparison between both approaches is very difficult and we are obliged to provide a fair comparison. Therefore, we initially summarize some differences between both approaches. Our approach is an offline training method and it takes advantages of the existence of external information systems. The classification model can be retrained during runtime while the online resources are not adversely affected. Therefore, the retrained model has to be transmitted from time to time to the online subcycle. The OVA-based multi-class classification model provides a very flexible structure while each class can be bounded by means of an optimized classifier. Hence, the time consumption depends on the selected one-class classifiers. Conversely, the HS-Tree approach is an online training method without any further transmission of the model. The HS-Tree approach intersects the state space by means of hyperplanes. Thus, classification results could be ambiguous if the classes are not paraxially separable.

However, the publication [29] claims two contrary statements. First [29, p. 1511], the HS-Tree "... requires only normal data for training [...] The model features an ensemble of random HS-Trees, and the tree structure is constructed without any data.". Second [29, p. 1513], "Once HS-Trees are constructed, mass profile of normal data must be recorded in the trees before they can be employed for anomaly detection.". Therefore, the HS-Tree approach is initially an offline training method while historical data can be used for training. It would be very negligent in some application domains to apply such an untrained and unassessed anomaly detecting model. On that score, we used the HS-Tree implementation as an offline training method while the model is online adaptive. This indicates another disadvantage of the HS-Tree. The HS-Tree is really big and many data needs to be transferred to the online subcycle which expenses a lot system resources (e.g. data path and time). Table 4 summarizes the processing times of the HS-Tree. The ensemble uses three trees for training. As can be seen, the processing time is approximately one millisecond smaller than our approach.

Table 4. HS-Tree classification time

$\mathcal{X}_1^{test}$ in ms	$\mathcal{X}_2^{test}$ in ms
5.08	5.34

7 Conclusion

The time-efficiency is very important for data stream processing and especially for data stream anomaly detection under resource restrictions. We suggest an OVA-based multi-class classification model for data stream anomaly detection. Moreover, we introduced classifier pre-selection as an extension of the OVA-based multi-class classification. We performed a real world case study and the results are very promising. The results show that the average processing time can reduced due to the selection of an optimized arrangement of the component one-class classifiers. The processing time can be further decreased by the use of classifier pre-selection. We compared our suggestions with the HS-Tree approach.

Acknowledgements

We wish to thank and acknowledge DLR, ESA and ASTRIUM Space Transportation for their insights and support, with special thanks to Enrico Noack. Moreover we would like to thank Swee Chuan Tan for the provision of the original HS-Tree source code. This work was supported by the Brandenburg Ministry of Science, Research and Culture as part of the International Graduate School at Brandenburg University of Technology Cottbus.

References

1. Babu, S., Widom, J.: Continuous Queries over Data Streams. SIGMOD Record 30(3), 109–120 (September 2001)
2. Bellmann, R.: Adaptive Control Processes. Princeton University Press (1961)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Journal of Machine Learning Research (JMLR) 11, 1601–1604 (2010)
4. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Data Stream Mining - A Practical Approach. Tech. rep., Centre for Open Software Innovation (COSI) - Waikato University (2011), http://heanet.dl.sourceforge.net/project/moa-datastream/documentation/StreamMining.pdf
5. Brereton, R.G.: One-Class Classifiers. Journal of Chemometrics 25(5), 225–246 (2011)
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection: A Survey. ACM Comput. Surv. 41, 15:1–15:58 (2009)
7. Cugola, G., Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Comput. Surv. 44(3), 15:1–15:62 (2012)
8. Dietterich, T.G.: Machine-Learning Research: Four Current Directions. AI Magazine 18, 97–136 (1997)
9. Domingos, P., Hulten, G.: Mining High-Speed Data Streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 71–80. KDD '00, ACM (2000)
10. Efron, B.: Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. Journal of the American Statistical Association 78(382), pp. 316–331 (1983)
11. Gama, J.: Knowledge Discovery from Data Streams. Chapman & Hall (2010)

12. Golab, L., Özsu, M.T.: Data Stream Management. Morgan & Claypool Publishers (2010)
13. Hashemi, S., Yang, Y., Mirzamomen, Z., Kangavari, M.: Adapted One-versus-All Decision Trees for Data Stream Classification. IEEE Transactions on Knowledge and Data Engineering 21(5), 624–637 (2009)
14. Hsu, C.W., Lin, C.J.: A Comparison of Methods for Multiclass Support Vector Machines. Neural Networks, IEEE Transactions on 13(2), 415–425 (2002)
15. Janssens, J.H.M., Flesch, I., Postma, E.O.: Outlier Detection with One-Class Classifiers from ML and KDD. In: Proceedings of the 2009 International Conference on Machine Learning and Applications. pp. 147–153. ICMLA '09, IEEE Computer Society (2009)
16. Knuth, D.E.: The Art of Computer Programming, Volume 3: Sorting and Searching. Addison Wesley Longman Publishing Co., Inc., 2nd ed. edn. (1998)
17. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
18. Li, C., Zhang, Y., Li, X.: OcVFDT: One-class Very Fast Decision Tree for One-class Classification of Data Streams. In: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data. pp. 79–86. SensorKDD '09, ACM (2009)
19. Noack, E., Belau, W., Wohlgemuth, R., Müller, R., Palumberi, S., Parodi, P., Burzagli, F.: Efficiency of the Columbus Failure Management System. In: AIAA 40th International Conference on Environmental Systems (2010)
20. Noack, E., Noack, T., Patel, V., Schmitt, I., Richters, M., Stamminger, J., Sievi, S.: Failure Management for Cost-Effective and Efficient Spacecraft Operation. In: Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). IEEE Computer Society (2011)
21. Noack, T., Schmitt, I.: Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle - A Case Study. In: Workshop on Knowledge Discovery, Data Mining and Machine Learning (KDML) (2012)
22. Noack, T., Schmitt, I.: Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle. In: Seventh IEEE International Conference on Research Challenges in Information Science (RCIS) (2013)
23. Raspberry Pi: Raspberry Pi (2013), http://www.raspberrypi.org/, online 2013-02-27
24. dos Santos Teixeira, P.H., Milidiú, R.L.: Data Stream Anomaly Detection through Principal Subspace Tracking. In: Proceedings of the 2010 ACM Symposium on Applied Computing. pp. 1609–1616. SAC '10, ACM (2010)
25. Seeger, M.: Learning with Labeled and Unlabeled Data. Tech. rep., University of Edinburgh (2000), http://lapmal.epfl.ch/papers/review.pdf
26. Smith, W.E.: Various Optimizers for Single-Stage Production. Naval Research Logistics Quarterly 3(1-2), 59–66 (1956)
27. Spinosa, E.J., de Leon F. de Carvalho, A.P., Gama, J.: Novelty Detection with Application to Data Streams. Intell. Data Anal. 13(3), 405–422 (2009)
28. Stonebraker, M., Çetintemel, U., Zdonik, S.: The 8 Requirements of Real-Time Stream Processing. SIGMOD Rec. 34(4), 42–47 (2005)
29. Tan, S.C., Ting, K.M., Liu, T.F.: Fast Anomaly Detection for Streaming Data. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two. pp. 1511–1516. IJCAI'11, AAAI Press (2011)
30. Tax, D.M.J.: One-Class Classification: Concept-learning in the Absence of Counter-Examples. Ph.D. thesis, TU Delft (2001), http://homepage.tudelft.nl/n9d04/thesis.pdf

31. Tax, D.M.J., Duin, R.P.W.: Using Two-class Classifiers for Multiclass Classification. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. vol. 2, pp. 124–127 (2002)
32. Tax, D.: DDtools, the Data Description Toolbox for Matlab (May 2012), http://prlab.tudelft.nl/david-tax/dd_tools.html, version 1.9.1
33. Tsai, J.J.P., Yang, S.J.H.: Monitoring and Debugging of Distributed Real-Time Systems. IEEE Computer Society Press (1995)